Fall 2016 - CMU OCI

25m : 17s
to test end

1
2
3
4
5

Element Present in Tree

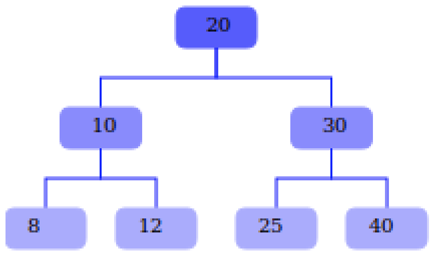
Each node of a *Binary Search Tree (BST)* has an integer *value* and pointers to two children, referred to as *left child* and *right child*. The value of *left child* is always less than the value of its parent node, and the value of *right child* is always greater than or equal to the value of its parent node.

The *isPresent* function in your editor has two parameters: a reference to the *root* node of a *Binary Search Tree (BST)* and an integer *value*. Complete *isPresent* so it returns *1* if the *value* is present in the *BST*, and returns *0* otherwise.

Constraints

- $1 \le total\ nodes \le 10^5$
- $1 \le value \le 5 \times 10^4$

Sample Input 0



Values

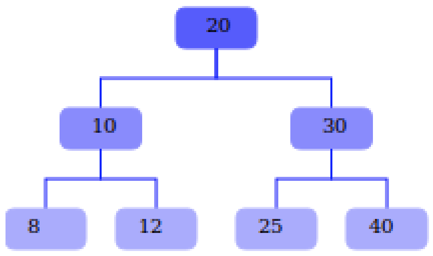
```
30
10
12
15
```

Sample Output 0

```
1
1
1
0
```

Value: *30*. This value is *present* in the *BST*, so *isPresent* returns *1*.
Value: *10*. This value is *present* in the *BST*, so *isPresent* returns *1*.
Value: *12*. This value is *present* in the *BST*, so *isPresent* returns *1*.
Value: *15*. This value is *not present* in the *BST*, so *isPresent* returns *0*.

Sample Input 1



Values

```
79
10
20
30
40
```

Sample Output 1

```
0
1
1
1
1
```

Explanation

Value: *79*. This value is *not present* in the *BST*, so *isPresent* returns *0*.
Value: *10*. This value is *present* in the *BST*, so *isPresent* returns *1*.
Value: *20*. This value is *present* in the *BST*, so *isPresent* returns *1*.
Value: *30*. This value is *present* in the *BST*, so *isPresent* returns *1*.
Value: *40*. This value is *present* in the *BST*, so *isPresent* returns *1*.

YOUR ANSWER

Fall 2016 - CMU OCI

25m : 17s
to test end

Start tour

1

2

3

4

5

Original code

Java 7

Fall 2016 - CMU OCI

25m : 17s
to test end

1
2
3
4
5

```
    private static class Node {
        Node left, right;
        int data;

        Node(int newData) {
            left = right = null;
            data = newData;
        }
    }

    private static Node insert(Node node, int data) {
        if (node==null) {
            node = new Node(data);
        }
        else {
            if (data <= node.data) {
                node.left = insert(node.left, data);
            }
            else {
                node.right = insert(node.right, data);
            }
        }
        return(node);
    }

    public static void main(String [] args) throws Exception{
        Scanner in = new Scanner(System.in);
        Node _root;
        int root_i=0, root_cnt = 0, root_num = 0;
        root_cnt = in.nextInt();
 _root=null;
for(root_i = 0; root_i < root_cnt; root_i++){
        root_num = in.nextInt();
        if( root_i == 0)
          _root = new Node(root_num);
        else
            insert(_root, root_num);
}

        int q = in.nextInt();

        for (int i = 0; i < q; i++) {
          int _x = in.nextInt();
          System.out.println(isPresent(_root,_x));
        }

```

Fall 2016 - CMU OCI

25m : 17s
to test end

1
2
3
4
5

```
private static int isPresent(Node root, int val){
/* For your reference
class Node {
        Node left, right;
        int data;
                Node(int newData) {
            left = right = null;
            data = newData;
        }
    }
*/
    return helper(root, val);
}

private static int helper(Node root, int val) {
    if (root == null) {
        return 0;
    }

    if (val == root.data) {
        return 1;
    } else if (val < root.data) {
        return helper(root.left, val);
    } else {
        return helper(root.right, val);
    }

}
}
```

Line: 57 Col: 1

Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

About Privacy Policy Terms of Service